Integrating the Quality Attribute Workshop (QAW) and the Attribute-Driven Design (ADD) Method

Robert L. Nord
William G. Wood
Paul C. Clements

July 2004

Software Architecture Technology Initiative

Technical Note
CMU/SEI-2004-TN-017

This work is sponsored by the U.S. Department of Defense.

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Contents

List of Figures

List of Tables

Abstract

The Software Architecture Technology Initiative at the Carnegie Mellon® Software Engineering Institute (SEI) has developed a number of architecture-centric methods that are currently in use. The initiative is now focusing on integrating these methods, as well as building bridges between them and software development processes and software architecture efforts outside the SEI, while continuing to refine existing methods and models. The goal is to provide software architects with a comprehensive, end-to-end approach for creating and using the right software architecture for the job at hand.

This technical note reports on a proposal to integrate the SEI Quality Attribute Workshop (QAW) and the SEI Attribute-Driven Design (ADD) method. The QAW is a way to elicit and articulate detailed quality attribute requirements for a system that an architecture must support. ADD is an architectural design method that starts with statements of quality attribute requirements and guides the architect through a series of design decisions that help to meet those requirements. Integrating these methods involves tailoring the QAW to provide the types of results needed by ADD and tailoring the ADD method to take full advantage of the results provided by the QAW.

1 Introduction

The Software Architecture Technology (SAT) Initiative at the Carnegie Mellon® Software Engineering Institute (SEI) has developed a number of architecture-centric methods. The Architecture Tradeoff Analysis Method® (ATAM®) helps a system's stakeholder community understand the consequences of architectural decisions with respect to the system's quality attribute requirements and business goals [Bass 03, Kazman 00]. The ATAM helps stakeholders ask the right questions to discover potentially problematic architectural decisions. The risks discovered from this process can then be made the focus of mitigation activities.

As members of the SAT Initiative gained experience from conducting ATAM evaluations, they developed methods that extend earlier into the software development life cycle. The SEI Quality Attribute Workshop (QAW) provides a method for eliciting quality attribute requirements [Barbacci 03]. The Attribute-Driven Design (ADD) method provides an approach for defining a software architecture by basing the design process on the system's quality attribute requirements [Bachmann 00].

Members of the initiative also developed complementary evaluation methods. SEI Active Reviews for Intermediate Designs (ARID) [Clements 02] are based on the ATAM. ARID concentrates on whether the design being proposed is suitable for other parts of the architecture that must use it. The SEI Cost Benefit Analysis Method (CBAM) is a method for architecture-based economic analysis of software-intensive systems [Bass 03, Kazman 02]. It can be used to help the system's stakeholders choose architectural alternatives for enhancing the system during the design or maintenance phases of the software development life cycle.

Members of the SAT Initiative started an effort to integrate the methods explicitly with each other and to integrate them into an organization's architecture-based software development life cycle, building on the common heritage, set of concepts, and activities the methods share. Previous reports showed how the architecture-centric methods can influence a wide variety of activities throughout the software development life cycle and how the method's activities can be distributed across a generic software development life cycle [Kazman 03], how the ATAM can be integrated with the CBAM [Nord 03], and how the architecture-centric methods are integrated into the Rational Unified Process (RUP) [Kazman 04].

® Carnegie Mellon, Architecture Tradeoff Analysis Method, and ATAM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

This report concentrates on the QAW and the ADD method to show how they can be enhanced and integrated within a software development life-cycle process to help organizations methodically design complex software-intensive systems. Such an architecture-centric development life cycle would include the activities shown in Figure 1 [Bass 03]. The integrated QAW/ADD elicits quality attribute scenarios to understand the architecturally significant requirements and creates the software architecture.

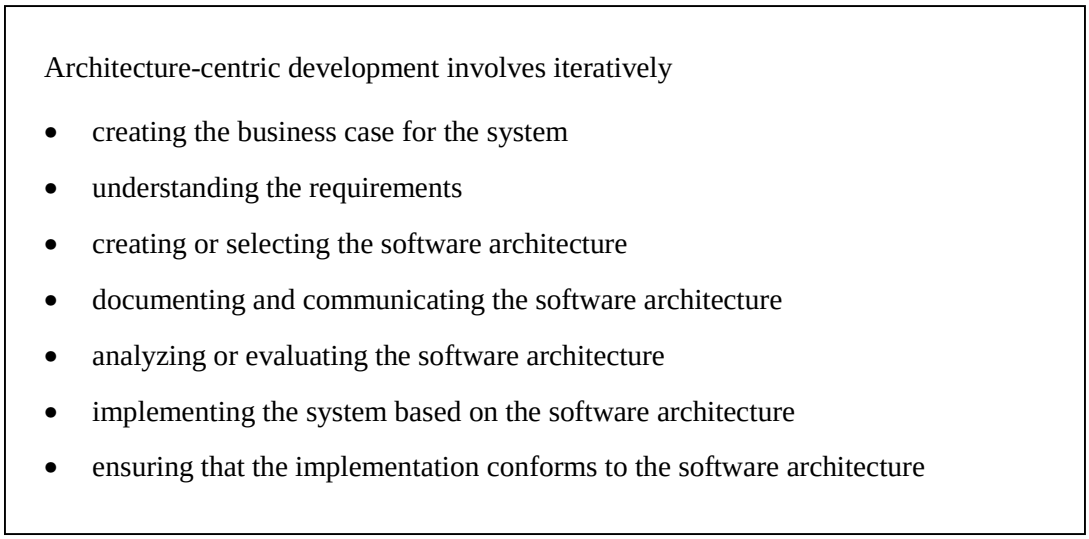

Figure 1: Life-Cycle Activities of Architecture-Centric Development

In Section 2 of this report, we demonstrate the need for integrating the QAW and the ADD method. In Section 3, we propose an integrated QAW/ADD method. In Section 4, we reflect on the integration issues and note opportunities for further work.

2 A Need for Integrating the QAW and the ADD Method

Assume that an organization is planning to develop a new system or evolving a system it has already fielded. The business drivers and a high-level description of the system have been documented. The organization needs help to understand the precise meaning of quality attributes—such as modifiability, security, performance, and reliability—in the context of the system being built. The organization also has responsibility for system development and needs help in designing an architecture that will enable the system to meet its quality goals.

This report is intended to help an architect or project manager meet these responsibilities through the use of a single method that embodies both the QAW and the ADD method. We are seeking to integrate the QAW and the ADD method, not merely append one to the other. The benefits of such an integrated method would be the combination of otherwise duplicative steps, more timely collection of necessary information, and more effective collection and use of that information to achieve the desired architectural outcomes.

2.1 The QAW

The QAW is a facilitated method that engages a diverse group of system stakeholders early in the life cycle to discover the driving quality attributes of a software-intensive system. The QAW provides a way to identify important quality attributes and the scenarios associated with them and to clarify system requirements *before* the software architecture has been created. The QAW elicits, collects, and organizes software quality attribute requirements in the form of scenarios. The results of the QAW are intended to be followed by an analysis and planning activity to determine further steps, such as applying the ADD method.

Figure 2 provides a summary of the inputs, outputs, and participants of the QAW. This figure is based on a functional modeling notation [IEEE 98] where inputs flow in from the left, outputs flow out to the right, and the participants of the method are noted below. When we refer to the QAW in this report, we're referring specifically to the third edition of the QAW as documented by Barbacci and associates [Barbacci 03]. Note that some of the suggestions in this report will be incorporated in revisions to the QAW.

Table 1 shows the steps for the QAW.

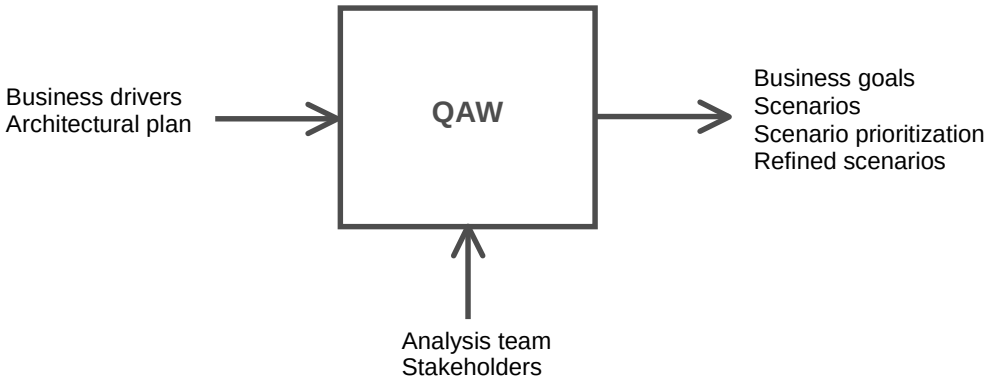

Figure 2: QAW Inputs, Outputs, and Participants

Table 1: The Steps of the QAW

QAW Steps	Description
Step 1 – QAW Presentation and Introductions	QAW facilitators describe the motivation for the QAW and explain each step of the method. Next, the facilitators introduce themselves and the stakeholders do likewise, briefly stating their background, their role in the organization, and their relationship to the system being built.
Step 2 – Business/Programmatic Presentation	A stakeholder representing the business and/or programmatic concerns presents the system's business/programmatic context, high-level functional requirements, constraints, and quality attribute requirements.
Step 3 – Architectural Plan Presentation	A technical stakeholder presents the system architectural plans including (1) plans and strategies for how key business/programmatic requirements will be satisfied; (2) key technical requirements, risks, and constraints—such as mandated operating systems, hardware, middleware, and standards—that will drive architectural decisions; (3) existing context diagrams, high-level system diagrams, and other written descriptions; (4) operational and system architectures, and architectural frameworks, tools, and architectural life-cycle processes being used; and (5) the prototyping and engineering studies underway to mitigate known risks.
Step 4 – Identification of Architectural Drivers	The facilitators share their list of key architectural drivers that include high-level requirements, business drivers, constraints, and quality attributes.
Step 5 – Scenario Brainstorming	The facilitators ask the stakeholders to brainstorm scenarios that are operationally meaningful with respect to the stakeholders' individual roles.
Step 6 – Scenario Consolidation	Similar scenarios are consolidated when reasonable.

Table 1: The Steps of the QAW (cont.)

QAW Steps	Description
Step 7 – Scenario Prioritization	Stakeholders vote to establish the priorities of the scenarios.
Step 8 – Scenario Refinement	The high-priority scenarios are refined in more detail. Facilitators further elaborate each one, documenting the following: the six parts of the scenario, the business/programmatic goals that are affected by this scenario, the relevant quality attributes associated with this scenario, and the questions and issues regarding the scenario.

2.2 The ADD Method

The ADD method defines a software architecture by basing the design process on the quality attributes that the software must fulfill; thus, it can complete the functional candidate architecture defined by the RUP. The ADD method documents a software architecture in a number of views: most commonly, a module decomposition view, a concurrency view, and a deployment view [Clements 03]. The ADD method depends on an understanding of the system's constraints and its functional and quality requirements. Figure 3 provides a summary of the inputs, outputs, and participants of the ADD method. The ADD method is targeted at software architects who develop the software architecture. Bass and colleagues provide more details about the ADD method [Bass 03].

Table 2 shows the steps for the ADD method.

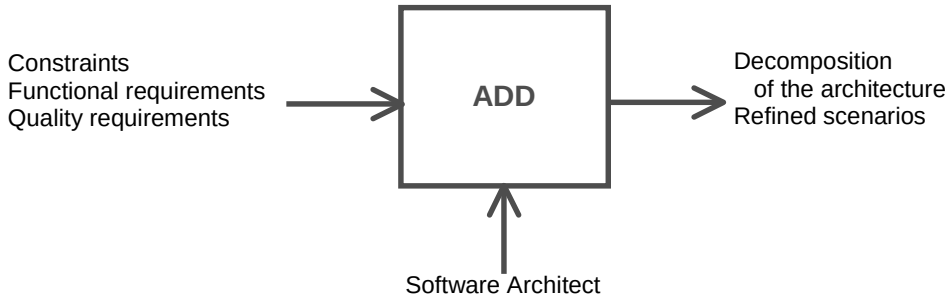

Figure 3: ADD Inputs, Outputs, and Participants

Table 2: The Steps of the ADD Method

ADD Steps	Description
Step 1 – Choose the module to decompose.	Use the whole system as the initial module to start ADD. All required inputs for this module should be available (constraints, functional requirements, and quality requirements).
Step 2 – Refine the module according to these steps:	
a – Choose the architectural drivers.	Choose the architectural drivers from the set of concrete quality scenarios and functional requirements. This step determines what is important for this decomposition.
b – Choose an architectural pattern that satisfies the architectural drivers.	Create or select the pattern based on the tactics that can be used to achieve the drivers. Identify the child modules required to implement the tactics.
c – Instantiate the modules and allocate functionality from the uses cases using multiple views.	Consider views from each of the three major groups of views (module decomposition, concurrency, and deployment).
d – Define the interfaces of the child modules.	The decomposition provides modules and constraints on the types of module interactions. Document this information in the interface documents for each module.
e – Verify and refine the use cases and quality scenarios, and make them constraints for the child modules.	This step verifies that nothing important was forgotten and prepares the child modules for further decomposition or implementation.
Step 3 – Repeat the steps above for the next module.	

2.3 Using the QAW and the ADD Method Together

As shown in Figures 2 and 3, the QAW provides business goals and a collection of quality attribute scenarios for the system that define the system's quality requirements, whereas the ADD method requires constraints, functional requirements, and quality requirements. The quality requirements are expressed as quality attribute scenarios for the system. In addition, the QAW is focused on eliciting information about the scenarios and quality attributes that drive the system as determined by a diverse group of stakeholders, whereas the ADD method is focused on using the QAW results as inputs for developing the software architecture by a group of architects.

Note that the architectural drivers identified during the QAW are an initial collection based on the facilitators' experience in distilling what they heard from the business and architectural plan presentations. The ADD method provides further guidance on refining the collection and choosing those drivers that will contribute to the selection of a major architectural style or pattern that will shape the architecture.

There is clearly a match between QAW outputs and ADD inputs. The QAWs conducted so far have all been for large-scale software-intensive distributed systems of systems[1] at an early stage of their development. They have resulted in scenarios that address more system-oriented issues, rather than software-oriented issues. Hence, the QAW results that address system-oriented issues will need to be transformed before they can be used directly in the ADD method. This transformation is discussed in Section 3.2.3.

[1] Systems of systems are built from components that are large-scale systems in their own right. Prominent examples include integrated air defense networks, the Internet, intelligent transport systems, and enterprise information networks. Systems of systems should be distinguished from large monolithic systems by the independence of their components, their evolutionary nature, emergent behaviors, and a geographic extent that limits the interaction of their components to information exchange [Maier 99].

3 An Integrated QAW and ADD Method

The integrated method is described in three portions: tailoring the QAW, bridging the QAW and the ADD method, and tailoring the ADD method.

3.1 Tailoring the QAW

Table 3 shows the changes to the QAW that are required when it is integrated with the ADD method. Terms used in the table are described in the text following the table. You may want to skim the table first to get a general understanding of the approach before reading the text and then return to the table for a summary of the approach. Steps not requiring changes are not discussed.

Our goal is to build on the success of the QAW and thus preserve the steps and objectives of the QAW, while at the same time tailoring its steps to optimize its integration with the ADD method.

Table 3: Tailoring the QAW

QAW Steps	Changes
Step 1 – QAW Presentation and Introductions	None
Step 2 – Business/ Programmatic Presentation	The business plan must include a table mapping the business goals to the quality attributes and a prioritization of this mapping.
Step 3 – Architectural Plan Presentation	The architect includes a list of the major risks and design challenges facing the system architects.
Step 4 – Identification of Architectural Drivers	The table mapping the business goals to the quality attribute drivers is expanded to include any new entries that arose during the architectural plan presentation.
Step 5 – Scenario Brainstorming	Scenarios are elicited that are software oriented and focused on the architect's design challenges as much as is reasonable; however, experience has shown that many of the scenarios will be system oriented. The facilitator ensures that the listed quality attributes are well covered by the scenarios and that the quality attributes and business goals associated with each scenario are captured.
Step 6 – Scenario Consolidation	None
Step 7 – Scenario Prioritization	After the voting, the architect "promotes" one of the lower level scenarios into the group to be refined.
Step 8 – Scenario Refinement	To aid ADD, the questions concentrate on the quality attribute aspects of the scenario and any concerns that the stakeholders might have in achieving the response called for in the scenario.

Step 2 – Business/Programmatic Presentation

The business plan presentation defines the business goals and includes a table showing how the quality attributes map to the business goals. The business plan also prioritizes the table entries. This mapping requires coordination among the engineers involved with business planning, the system and software architects, and those developing the operational concepts.

Step 3 – Architectural Plan Presentation

The architect should discuss process challenges faced by the architecture team, such as the following: the degrees of freedom that are still open in the system architecture and the operational concepts, how to choose the appropriate software architecture views, the tool set to be used to build the architecture, the process for evaluating the architecture, the relationship to system and operational views, and studies and prototypes already underway (and planned) to mitigate the identified risks.

Step 4 – Identification of Architectural Drivers

The facilitators share their list of key architectural drivers that they have assembled from listening to the presentations so far. These drivers include high-level requirements, business drivers, constraints, and quality attributes. The architect will also include a list of the architectural challenges presented by the system, to give the stakeholders some concept of the difficulties faced by the architecture team, and map each challenge to the associated quality attribute concerns.

Step 5 – Scenario Brainstorming

The QAW participants will be able to see the tables showing the business goals versus quality attribute concerns and the architectural challenges versus quality attribute concerns. As each scenario is generated, the mappings that have been addressed by the scenario will be recorded.

During this step, the facilitator will refer to the previously mentioned tables in an effort to cover them reasonably with at least one scenario for each entry. Although scenarios in the QAW are explicitly generated using a bottom-up brainstorming technique, the facilitator implicitly relies on the notion of a utility tree to ensure coverage.

A one-page sheet will be passed out that explains the format of the scenarios and has some blank space to allow the participants to prepare their scenarios.

Step 7 – Scenario Prioritization

Stakeholders are allowed to cast their votes in the manner of their choosing. When the voting is completed, the architect (in consultation with the business person) will promote one of the lower level scenarios to be refined.

Step 8 – Scenario Refinement

The collection of general scenarios [Bass 03] provides a way to characterize attributes and guide scenario refinement. The general scenario tables will be passed out and used to fill in the parts of the scenario in the refinement table; in doing so, the relevant general scenarios are made system specific.

The participants will be encouraged to raise questions and issues associated with the architectural challenges.

3.2 Bridging the QAW and the ADD Method

To prepare for applying the ADD method, we recommend performing the following activities:

1. Analyze the QAW results.
2. Hold a post-QAW planning workshop.
3. As necessary, transform the elicited scenarios so they're useful for ADD.

The results should be analyzed immediately after the QAW, and the post-QAW planning workshop should follow that analysis. The transformation of the scenarios can occur during the post-QAW planning workshop.

Other requirements analysis occurs in parallel with the QAW; for example, refinement of the functional requirements into the form of use cases. Scenarios generated during the QAW could lead to multiple use cases.

3.2.1 Analyze the QAW Results

At the completion of the QAW, information associated with the quality attributes has been generated and prioritized, but it has not been put into context. The first step toward providing this context is to analyze the QAW results. The suggested steps for this analysis are described below:

1. Build a utility tree showing how the scenarios are organized according to the quality attributes.

2. Build a table showing how the scenarios map to the business goals, and identify any missing business goals that are obvious from the QAW results.
3. Build a table showing how the scenarios map to the architectural drivers, and identify any missing architectural drivers.
4. Build a table showing the architectural life-cycle processes being used and how they relate to the scenarios. Identify any missing processes implied by a scenario.
5. Develop a list of concerns resulting from the missing business goals, architectural drivers, and architectural life-cycle processes. Note any concerns resulting from the lack of scenario coverage in any of the business goals, architectural drivers, and processes identified earlier.
6. Identify any potential studies, and prototype development needed to mitigate the risks and concerns that have been identified.

3.2.2 Hold a Post-QAW Planning Workshop

After the QAW results have been analyzed, a post-QAW planning workshop can be held to review the concerns expressed in the results analysis and determine what further actions can be taken to assist in the ADD approach. Suggested activities for such a workshop are listed below:

1. Review the concerns and classify them in some manner: for example, those that can be dealt with in a cursory manner (e.g., by adding a new business goal), those that require some further exploration (e.g., by writing appropriate white papers or doing feasibility studies), and those that require extensive integration with outside agencies.
2. Determine if any of the lower priority scenarios need to be refined, and then refine them.
3. Perform flow downs of the scenarios (see Section 3.2.3).

3.2.3 As Necessary, Transform the Elicited Scenarios So They're Useful for ADD

Some scenarios from QAW may be useful directly in ADD. Others may need to be refined and allocated to hardware, software, people, or data in the system architecture. The ADD method can be used on the software-architecture portion of the system architecture.

The QAW stakeholders often express high-level scenarios of the following types:

- operational scenarios. These scenarios describe, for example, how a system of systems responds to a stimulus while under a heavy workload. These scenarios are often at the following levels: mission or business threads (stimulus to response); resource changes such as operators, communications, sensors, and hardware and software components; conditions under which the data integrity, data consistency, and security of access must be maintained; interoperability between different releases of the system; or switching from training mode to operational mode.
- life-cycle process scenarios. These scenarios are concerned largely with the life-cycle processes involved with the system of systems. Life-cycle scenarios often describe detecting and isolating software errors during operation and in integration testing, building and releasing systems in blocks, installing new systems and upgrades in the field, including commercial off-the-shelf (COTS) products developed from software standards, or making modifications to the system to enhance its operation.

In many large-scale software-intensive systems, there is a significant body of existing documents, such as the following: the system requirements document (SRD); the concept of operations (CONOPS) document or operational views describing the CONOPS; system scenarios describing the activities to be performed by the end users, the interrelationships among the activities, and the relationships between these activities and individual systems; and some appropriate system architectural views.

For example, a QAW scenario may not distinguish between what is done by operators and what is done by the automation, especially where multiple operators are involved in the scenario. In this case, the architect will have to review the existing documentation (such as those documents listed above) and transform the original scenario into a group of overlapping scenarios that combine to provide the original scenario. Some examples of such transformations are given below:

- A system-specific quality scenario describes the operation between a sensor detecting an event of interest and an actuator responding to the event, including a time deadline for the response; however, it ignores the manual operations involved. The architect reviews the other documentation and discovers that the scenario involves a number of operators at a number of locations using different systems. The architect must then re-express this scenario as a number of scenarios that can be used to develop the software architecture for the interoperation between the systems, allocating timing budgets to each scenario to satisfy the original timing deadline.
- A system-specific quality scenario describes an activity involving many (hundreds of) nodes connected by a low bandwidth communication system. The software architect knows enough from the scenario to separate out the software architectural decisions from the system decisions and to build a software architectural fragment to satisfy the scenario. However, the architect cannot verify that the scenario will be satisfied until a simulation study is executed for the environmental workload included in the scenario. In this case, he or she will probably have to work cooperatively with the system engineers.

- A 30-day quick update requirement is levied on the system to correct some inadvertent behavior. The software architect must then describe how this requirement affects the architecture, development, integration, test, and field-deployment organizations.

The ADD method has an activity (Step 2e) for refining use cases, constraints, and scenarios that could be applicable here. Requirements and constraints flow from the module chosen to decompose to its child modules. Quality scenarios also have to be refined and assigned to the child modules. This is done in one of the following ways:

- A quality scenario may be completely satisfied by the decomposition without any additional impact. It can then be marked as satisfied.
- A quality scenario may be satisfied by the current decomposition with constraints on child modules.
- The decomposition may be neutral with respect to a quality scenario. This scenario should be assigned to one of the child modules.
- A quality scenario may not be satisfied with the current decomposition. If the scenario is important, the decomposition should be reconsidered. Otherwise, the rationale for the fact that the decomposition does not support this scenario must be recorded. This occurrence is usually the result of a tradeoff with other, perhaps higher priority, scenarios.

The Rational Unified Process for Systems Engineering (RUP SE) has a notion of use case flow down that could be applicable here [Cantor 03].

3.3 Tailoring the ADD Method

Table 4 shows the steps for a tailored ADD method, as we envision it, that could be applied when it is integrated with the QAW. The steps that are changed as a result of tailoring are described in more detail after the table. Steps not requiring changes are not discussed.

Table 4: Tailoring the ADD Method

ADD Steps	Changes
Step 1 – Choose the module to decompose.	None
Step 2 – Refine the module according to these steps:	
a – Choose the architectural drivers.	This step is completed by the QAW and the Analyze the QAW Results activities for the first iteration of the ADD method (where the module chosen in Step 1 is the system).
b – Choose an architectural pattern that satisfies the architectural drivers.	The QAW begins the process of “type-checking” the scenarios.

Table 4: Tailoring the ADD Method (cont.)

ADD Steps	Changes
c – Instantiate the modules and allocate functionality from the uses cases using multiple views.	None
d – Define the interfaces of the child modules.	None
e – Verify and refine the use cases and quality scenarios, and make them constraints for the child modules.	None
Step 3 – Repeat the steps above for the next module.	

Step 2a – Choose the architectural drivers.

An architecture is shaped by some collection of functional (e.g., training crews in flight simulation), quality (e.g., high security), and business (e.g., product line) requirements. The ADD method refers to these shaping requirements as architectural drivers.

The architectural drivers must be expressed ultimately as quality attribute scenarios for the system.

ADD defines a three-step process for finding the drivers:

1. Identify the business goals with the highest priority.
2. Turn business goals into quality attribute scenarios of use cases (e.g., via construction of a utility tree).
3. Choose the quality attribute scenarios with the most impact on the architecture. These scenarios are the architectural drivers, and there should be fewer than 10.

The QAW-prioritized scenarios provide a pool of potential drivers from which to choose. Making enhancements to the QAW to record marketers' and architects' voting narrows the pool to those drivers that have the highest priority business goals and the most impact on the architecture. For the first iteration of the ADD where the module chosen in Step 1 is the system, this step is already completed by the QAW and the Analyze the QAW Results activity to construct a utility tree.

Step 2b – Choose an architectural pattern that satisfies the architectural drivers.

Bachmann, Bass, and Klein [Bachmann 03] provide more details for this step. Scenarios are "type-checked" by using general scenarios to fill in and identify the six parts of the concrete scenario. This, in turn, helps identify the associated reasoning frameworks and later the tactics.

Scenarios could be "type-checked" during the QAW refinement step. The general-scenario-generation tables can be used to help fill in the concrete scenario. At the same time, the general scenario can be generated (or, at least, the entries chosen from the table can be noted to facilitate later generation of the general scenario) in preparation for this ADD step.

Associated tactics might also be identified during the QAW to help with the generation of questions and issues.

4 Reflections on Integrating the QAW and the ADD Method

4.1 Aligning the QAW and the ADD Method

To integrate the QAW and the ADD method, we used our understanding of their artifacts and activities to align the two methods more closely.

The ADD method informs the QAW as described below:

- The information needed by the ADD method could constrain the QAW in terms of the types of scenarios generated, the stakeholders invited, the questions or issues recorded in the refinement template, and so forth.
- The notion of general scenarios was originally developed in the context of ADD. Incorporating them into the QAW helps guide scenario refinement into six parts.
- Step 2e of the ADD method might be applied to refine the system scenarios that flow from the QAW to the ADD method.

The QAW informs the ADD method as described below:

- An enhanced QAW can help guide or take the place of ADD Step 2a, in which the architectural drivers are chosen. Mapping quality attributes to business goals is necessary to support the construction of a utility tree before applying the ADD method.
- High-priority scenarios from the QAW inform the types of views that are selected, which encourages designers to look at the difficult design problems first.
- In additional to quality requirements, the ADD method also takes constraints and functional requirements as inputs. QAW scenarios could lead to use cases, which are input into ADD as functional requirements. Issues and questions in refined scenarios that are further refined into themes and suggestions could provide input into ADD as constraints.

4.2 Application to Small-Scale Systems

The basic intent of the QAW is to first lay out the intent of the system (by presenting business drivers and architectural drivers) and then to have the (10 to 25) *diverse stakeholders* of the system generate (30 to 40) scenarios from their viewpoint, prioritize them, and finally refine the (5 or so) high-priority scenarios further by capturing a number of questions and statements about each one.

The key reason for requiring a workshop to generate and prioritize scenarios is the diversity of the stakeholders. And such diversity almost always exists for large-scale systems. However, for smaller scale systems, the stakeholder diversity may not be great, and the architects may have sufficient experience to act as surrogate stakeholders. Hence, it may be possible to achieve the same results as a QAW in a smaller scale setting by getting the design team together in a room and performing the following activities:

1. Have the marketing representative present a few slides on the business goals.
2. Have the architect present a few slides on the architectural drivers, including the quality attributes of concern.
3. Have the team members generate scenarios for the system and then prioritize them.
4. Finally, refine the high-priority scenarios.

Doing this would produce the same results as the QAW, which could then be used as input for the ADD.

4.3 Future Work

Further work remains to be done to verify the benefits of our proposed integration approach through pilot projects with customers. The goal of such projects is to provide tailored architecture methods to help customers add architecture-based and quality-attribute-based thinking to their planning and development efforts.

Further work should also be done in improving techniques such as scenario flow down, bridging the boundary between system and software architectures, and extending the ADD method to the systems-of-systems or enterprise level.

We have noted in the previous section where the QAW and the ADD method interact seamlessly in the application to small-scale systems. QAW scenarios provide (directly or with minimal processing) architectural drivers for the ADD method. Further work remains in characterizing this problem set more precisely. This further work could put constraints on the kinds of systems to which this integrated method is applied and how it is applied. Targeting the QAW portion of the method to information needed by the ADD could influence the kinds of scenarios that are generated to focus on those that (1) are more software oriented than system oriented, (2) highlight major software design challenges, and (3) are used directly as inputs to the ADD method without further processing.

5 Summary

This technical note reports on a proposal to integrate the QAW and the ADD method. The QAW is a way to elicit and articulate detailed quality attribute requirements for a system that an architecture must support. ADD is an architectural design method that starts with statements of quality attribute requirements and guides the architect through a series of design decisions that help to meet these requirements. As such, these two methods are good candidates for method integration. The benefits of such an integrated method would be the combination of otherwise duplicative steps, more timely collection of necessary information, and more effective collection and use of that information to achieve the desired architectural outcomes.

The integration of different methods and techniques with each other or with other development life cycles is possible and will be addressed in future reports. Investigating such potential combinations is part of a larger effort to understand how to

- integrate the approaches
- package the methods' activities into the software development life cycles of typical organizations
- understand the appropriate fit with other architectural processes and technologies
- connect with other business, management, or acquisition processes that can help enforce software architecture practices throughout the life cycle
- make the methods more usable to software practitioners

References

URLs valid as of the publication date of this document

[Bachmann 00] Bachmann, F.; Bass, L.; Chastek, G.; Donohoe, P.; & Peruzzi, F. *The Architecture Based Design Method* (CMU/SEI-2000-TR-001, ADA375851). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. http://www.sei.cmu.edu/publications/documents/00.reports/00tr001.html

[Bachmann 03] Bachmann, F.; Bass, L.; & Klein, M. *Deriving Architectural Tactics: A Step Toward Methodical Architectural Design* (CMU/SEI-2003-TR-004, ADA413644). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03tr004.html

[Barbacci 03] Barbacci, M. R.; Ellison, R.; Lattanze, A. J.; Stafford, J. A.; Weinstock, C. B.; & Wood, W. G. *Quality Attribute Workshops (QAWs), Third Edition* (CMU/SEI-2003-TR-016, ADA418428). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html

[Bass 03] Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice, Second Edition*. Boston, MA: Addison-Wesley, 2003.

[Cantor 03] Cantor, M. “Rational Unified Process for Systems Engineering.” *Rational edge: the e-zine for the rational community* (September 2003). http://www-106.ibm.com/developerworks/rational/library/content/RationalEdge/sep03/m_systemarch_mc.pdf

[Clements 02] Clements, P.; Kazman, R.; & Klein, M. *Evaluating Software Architectures: Methods and Case Studies.* Boston, MA: Addison-Wesley, 2002.

[Clements 03] Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; & Stafford, J. *Documenting Software Architectures: Views and Beyond*. Boston, MA: Addison-Wesley, 2003.

[IEEE 98] Institute of Electrical and Electronics Engineers. *IEEE Standard for Functional Modeling Language* (IEEE Std 1320.1-1998). New York, NY: IEEE Computer Society, 1998 (ISBN 0-738-10340-3).

[Kazman 00] Kazman, R.; Klein, M.; & Clements, P. *ATAM: Method for Architecture Evaluation* (CMU/SEI-2000-TR-004, ADA382629). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. http://www.sei.cmu.edu/publications/documents/00.reports/00tr004.html

[Kazman 02] Kazman, R.; Asundi, J.; & Klein, M. *Making Architecture Design Decisions: An Economic Approach* (CMU/SEI-2002-TR-035, ADA408740). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. http://www.sei.cmu.edu/publications/documents/02.reports/02tr035.html

[Kazman 03] Kazman, R.; Nord, R. L.; & Klein, M. *A Life-Cycle View of Architecture Analysis and Design Methods* (CMU/SEI-2003-TN-026, ADA421679). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon Institute, 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03tn026.html

[Kazman 04] Kazman, R.; Kruchten, P.; Nord, R. L.; Tomayko, J. E. *Integrating Software-Architecture-Centric Methods into the Rational Unified Process* (CMU/SEI-2004-TR-011). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/publications/documents/04.reports/04tr011.html

[Maier 99] Maier, M. W. “Architecting Principles for Systems-of-Systems.” *Systems Engineering 1*, 4 (Feb. 1999): 267-284.

[Nord 03] Nord, R. L.; Barbacci, M. R.; Clements, P.; Kazman, R.; Klein, M.; O’Brien, L.; & Tomayko, J. E. *Integrating the Architecture Tradeoff Analysis Method (ATAM) with the Cost Benefit Analysis Method (CBAM)* (CMU/SEI-2003-TN-038, ADA421615). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03tn038.html

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	July 2004	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Integrating the Quality Attribute Workshop (QAW) and the Attribute-Driven Design (ADD) Method	F19628-00-C-0003

6. AUTHOR(S)

Robert L. Nord, William G. Wood, Paul C. Clements

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2004-TN-017

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

The Software Architecture Technology Initiative at the Carnegie Mellon® Software Engineering Institute (SEI) has developed a number of architecture-centric methods that are currently in use. The initiative is now focusing on integrating these methods, as well as building bridges between them and software-development processes and software-architecture efforts outside the SEI, while continuing to refine existing methods and models. The goal is to provide software architects with a comprehensive, end-to-end approach for creating and using the right software architecture for the job at hand.

This technical note reports on a proposal to integrate the SEI Quality Attribute Workshop (QAW) and the SEI Attribute-Driven Design (ADD) method. The QAW is a way to elicit and articulate detailed quality attribute requirements for a system, which the architecture must support. ADD is an architectural design method that starts with statements of quality attribute requirements and guides the architect through a series of design decisions that help to meet those requirements. Integrating these methods involves tailoring the QAW to provide the types of results needed by ADD and tailoring the ADD method to take full advantage of the results provided by the QAW.

14. SUBJECT TERMS	15. NUMBER OF PAGES
architecture-centric methods, Architecture Tradeoff Analysis Method, ATAM, Attribute-Driven Design method, ADD method, integrated QAW/ADD, Quality Attribute Workshop, QAW	30

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL